Modélisation et base de données
VII) SQL

Jérôme Fortin

Polytech'Montpellier
Université de Montpellier

2015-2016

Structured Query Language (SQL)

- le language SQL sert à manipuler les données ou la structure de la base de données
- Élaboré dans les années 70
- SQL est composé de trois partie majeures et distinctes :
  - Data Manipulation Language
  - Data Definition Language
  - Data Contol Language

Structured Query Language (SQL)

Principaux ordre employés :

DML (manipulation)	**DDL** (definition)	**DCL** (contrôle)
SELECT	CREATE	GRANT
INSERT	ALTER	REVOKE
DELETE	RENAME	
UPDATE	DROP	

Structured Query Language (SQL)

Une requête SQL peut être

- employée seule
- dans un éditeur de requête fourni par le SGBD
- directement intégrée dans un langage de programation

Exemple : une requête peut être testée directement avec phpMyAdmin, puis intégrée dans une page web grâce au langage PHP pour interagir depuis une page internet sur la base de donnée.

SELECT

La commande *SELECT* permet de réaliser une recherche d'information selon certains critères

Syntaxe

SELECT {ALL / DISTINCT} nom_attribut1 [, nom_attribut2,...]
FROM nom_table [, nom_table2,...]
WHERE <condition de recherche >

SELECT

- l'option **ALL** permet de selectionner l'ensemble des lignes satisfaisant la condition de recherche
- l'option **DISTINCT** permet de ne conserver que des lignes distinctes
- La **liste des attributs** indique la liste des colonnes choisies
- On utilise * pour selectionner l'ensemble des colonnes d'une table
- la **liste des tables** indique les tables sur lesquelles portent les opérations
- la **condition de recherche** permet d'exprimer des critères de recherche complexes à l'aide d'opérateurs logiques et/ou arithmétiques

Exemples

$T\_Clients(\underline{NumCli}, Nom, Prenom, Adresse, CodeP, Ville, Tel)$

$T\_Achats(\#\underline{NumCli}, \#\underline{NumArt}, Date, Qte)$

$T\_Articles(\underline{NumArt}, Designation, Categorie, Prix)$

Exemples

Table **T_Clients**

NumCli	Nom	Prénom	Adresse	CodeP	Ville	Tel
1	Fortin	Jérôme	2 r bonnard	34000	Montpellier	04 73 80 25
2	Fonglias	John	4 r Caillaud	44000	Nantes	
3	Fonglias	Céline	Av des Paulines	63 000	Clermont Fd	
4	Durand	Camille	Place d'espagne	75 019	Paris	01 45 78 25
5	Roger	Julie	16 r substantion	34000	Montpellier	04 75 24 63

Table **T_Achats**

#NumCli	#NumArt	Date	Qte
1	1	30/01/2009	1
1	5	30/01/2009	4
4	3	29/01/2009	1
4	2	30/01/2009	2
5	2	30/01/2009	2

Table **T_Articles**

NumArt	Désignation	Categorie	Prix
1	Product placement	Cd	12
2	Kaamelott	DVD	19
2	WebCam	Informatique	24
4	Graveur	Informatique	38
5	Clé USB 16 Go2	Informatique	18

Projection

Construire une relation en ne gardant que certains attributs

- Afficher le contenu de la table Clients

```
SELECT *
From T_Clients;
```

- Afficher le nom et prénom des clients

```
SELECT Nom, Prenom
From T_Clients;
```

Mot clé `Distinct`

Conserver que les lignes distinctes

Mot clé `All` actif par défaut

- Afficher les noms des Clients (avec doublons eventuels) :

  ```
  SELECT Nom
  From T_Clients;
  ```

- Afficher le nom et prénom des clients

  ```
  SELECT DISTINCT Nom
  From T_Clients;
  ```

Modification des entêtes affichées : As

```
SELECT DISTINCT Nom As "Nom des clients"
From T_Clients;
```

Restrictions : `Where`

- **Pour ne garder que les tuples vérifiants une certaine condition**
- Dans la clause `WHERE` les opérateurs suivants peuvent être utilisés :
  - comparateur numériques : $>$, $>=$, $<$, $<=$, $=$, $<>$
  - opérateurs logiques : `AND`, `OR`, `NOT`, `NOT NULL`, `ALL`, `ANY`, `EXISTS`...
  - IN BETWEEN LIKE

Restrictions : WHERE

- Clients habitant Montpellier

```
SELECT *
From T_Clients
WHERE Ville='Montpellier';
```

- Articles dont le prix est compris entre 14 et 30 Euros

```
SELECT *
From T_Articles
WHERE prix>14 AND prix < 30;
From T_Articles
WHERE prix BETWEEN 14 AND 30;
```

- Clients dont le numero est saisie

```
SELECT *
From T_Clients
WHERE Tel IS NOT NULL;
```

LIKE

Permet d'utiliser des expressions régulières

- Clients dont le nom commence par F

```
SELECT *
From T_Clients
WHERE Nom Like 'B%';
```

- Clients dont le nom fini par d

```
SELECT *
From T_Clients
WHERE Nom Like '%d';
```

- Clients dont le nom contient un F

```
SELECT *
From T_Clients
WHERE Nom Like '%B%';
```

Les tris

La clause `ORDER BY`, associée aux mots `ASC` ou `DESC` permet de trier les résultats :

```
SELECT *
FROM Clients
ORDER BY Nom DESC
```

NumCli	Nom	Prénom	Adresse	CodeP	Ville	Tel
4	Durand	Camille	Place d'espagne	75 019	Paris	01 45 78 25
3	Fonglias	Céline	Av des Paulines	63 000	Clermont Fd	
2	Fonglias	John	4 r Caillaud	44000	Nantes	
1	Fortin	Jérôme	2 r bonnard	34000	Montpellier	04 73 80 25
5	Roger	Julie	16 r substantion	34000	Montpellier	04 75 24 63

Les jointures

La **jointure** consite à **rapprocher deux relations** R1 et R2 afin de former une troisième relation R3 vérifiant une certaine **condition de rapprochement**.

- Afficher les noms et prénoms des clients ayant acheté un article le 30 février 2009

```
 SELECT Nom, Prenom
FROM Clients, Achats
WHERE Clients.NumClient=Achets.NumClient
AND Achat.Date='30/12/2009'
SELECT Nom, Prenom
FROM Clients JOIN Achats ON
Clients.NumClient=Achets.NumClient
WHERE Achat.Date='30/12/2009'
```

AS

Les noms de tables peuvent devenir fastidieux à saisir, on peut simplifier l'écriture en utilisant le mot clé **AS**.

- Afficher les noms et prénoms des clients ayant acheté un article le 30 février 2009

```
 SELECT Nom, Prenom
FROM Clients AS Cl JOIN Achats AS Ach ON
Cl.NumClient=Ach.NumClient
WHERE Achat.Date='30/12/2009'
```

Les fonctions statistiques

Fonction	**description**
AVG(attrubut)	Caul la moyenne des valeurs dans l'attribut
SUM(attribut)	Calcul la somme des valeurs dans l'attribut
MIN(attribut)	Détermine la plus petite valeur dans l'attribut
MAX(attribut)	Détermine la plus grande valeur dans l'attribut
COUNT(attribut)	Compte de nombre d'occurence dans l'attrubut

Calculer le prix moyen des articles :
SELECT AVG(Prix) As Prix moyen, MIN(Prix), MAX(Prix)
FROM Articles ;
Compter le nombre de cathégories : SELECT
COUNT(Cathégorie) As Nombre de cathégories
FROM Articles ;

Sous-requêtes

- Une sous requête permet de réaliser de façon élégante des traitement qui seraient fasitdieux voir difficilement réalisables à l'aide de requetes simples.
- le rôle d'une sous requête est d'envoyer le résultat de son traitement dans la requête principale

Articles dont le prix est suppérieur au prix moyen :
SELECT * FROM articles
WHERE Prix > (Select AVG(PRIX) FROM Articles);